Reto 1-20202110

Req. 2- Felipe Garzon Torres, f.garzon@uniandes.edu.co, 202021161

Req. 3- Pablo Ortega Cadavid, p.ortegac@uniandes.edu.co, 202021700

Analisis de complejidad:

Req 1

El primer recorrido que se hace para el requerimiento 1 tiene el propósito de crear una lista de menor tamaño para reducir el tiempo que se tarda la función de ordenamiento merge sort. Este primer recorrido tiene una complejidad de O(N). Después el merge sort tiene una complejidad de O(NlogN) pero con los N datos que proporciona la sublista que creamos previamente. Hacer la función de esta forma permite reducir el tiempo que se demora, ya que es favorable que cuando se haga el sort la cantidad N de datos sea la menor debido a que es la parte del requerimiento con mayor complejidad temporal. La complejidad total de la función podría variar desde aproximadamente O(N) a aproximadamente N log N dependiendo del tamaño del subconjunto que se cree con la primera función. Si la cantidad de datos en la sublista es cercana a N la complejidad será cercana a O(N log N) y si la cantidad de datos en la sublista es cercana a 0 entonces la complejidad será cercana a O(N)

Req 2-Felipe

Para este requerimiento se establecen dos diccionarios vacíos, una variable para hacer una comparación posterior y una variable como posicionador. Luego de esto para hacer el recorrido se usa un while con base al posicionador, con complejidad O(N). Luego de estos se establecen distintos filtros (if) para tener en cuenta solo los videos del país solicitado. Una vez se pasa el primer filtro, se pasa a un segundo donde se establece como llave de un diccionario el título, el canal y el país del video y de valor se usa una array-llist que contiene las fechas en las que el video estuvo de tendencia. Luego salido del recorrido, se establece otro recorrido con un for y complejidad de O(N) para determinar el video con mayor tendencia. La complejidad general del requerimiento es O(2N) lo que se aproxima a O(N).

Req 3-Pablo

Para el requerimiento se hace un primer recorrido que tiene complejidad O(N) que tiene como propósito leer el catálogo y organizar la información en un diccionario de array_lists creado previamente. Después se hace otro recorrido para encontrar video con más días en tendencias, este segundo recorrido también tiene complejidad O(N). Al sumar estas complejidades obtenemos que la complejidad temporal del requerimiento es O(2N) lo que se aproxima a O(N).

Req 4

Lo primero que se hace el algoritmo del requerimiento 4 es un merge sort que tiene una complejidad temporal media de O(N log N). Después se hace un recorrido de complejidad O(N). Esto significa que la complejidad general del requerimiento es O(2N log N) que se aproxima a O(N log N). Sin embargo, esta no es la menor complejidad posible para este punto. Es posible alcanzar una complejidad temporal cercana a O(N) si se filtran los datos antes de hacer el sort.